# STA355 Homework 3

Luis Rojas

21/03/2021

(c) The file `bees.txt` contains dance directions of 279 honey bees viewing a zenith patch of artificially polarized light. (The data are given in degrees; you should convert them to radians. The original data were rounded to the nearest $10^o$; the data in the file have been "jittered" to make them less discrete.) Using these data, compute the posterior density of $\kappa$ in part (b) for $\lambda = 1$ and $\lambda = 0.1$. How do these two posterior densities differ? Do these posterior densities "rule out" the possibility that $\kappa = 0$ (i.e. a uniform distribution)?
**Note:** To compute the posterior density, you will need to compute the normalizing constant – on Quercus, I will give some suggestions on how to do this. A simple estimate of $\kappa$ is

$$\widehat{\kappa} = \frac{\frac{r}{n}\left(2 - \frac{r^2}{n^2}\right)}{1 - \frac{r^2}{n^2}}$$

where $r$ is as defined in part (b). The posterior density should be largest for values of $\kappa$ close to $\widehat{\kappa}$; you can use this as a guide to determine for what range of values of $\kappa$ you should compute the posterior density.

```
bees <- scan("bees.txt")
```

**(The data are given in degrees; you should convert them to radians. The original data were rounded to the nearest 10degree ; the data in the file have been "jittered" to make them less discrete.)**

**Using these data, compute the posterior density of $\kappa$ in part (b) for $\lambda = 1$ and $\lambda = 0.1$. How do these two posterior densities differ? Do these posterior densities "rule out" the possibility that $\kappa = 0$ (i.e. a uniform distribution)?**

From b we have:

$$\pi(k|d_1, ..., d_n) \;=\; c(d_1, ..., d_n)\frac{exp(-\alpha\kappa)I_o(r\kappa)}{[I_o(\kappa)]^n}$$

The von Misses Distribution whose density on $[0, 2\pi]$ is:

$$f(\theta; \kappa, \mu) = \frac{1}{2\pi I_0(\kappa)}\; exp(\kappa\; cos(\theta - \mu))$$

We know the likelihood (considering only $\kappa$) has the form:

$$L(D_i\; ; \kappa, \mu) = (\frac{1}{2\pi})^n(\frac{1}{\prod_{i=1}^n I_0(\kappa)}) \int_0^{2\pi} exp\; [\sum_{i=1}^n\; \kappa\; cos(D_i - \mu)]\; d\mu$$

Writing in terms of the Bessel function of the first kind:

$$L(D_i\; ; \kappa, \mu) = \frac{1}{(2\pi)^{n-1}}\; \frac{1}{\prod_{i=1}^n I_0(\kappa)}\; \frac{1}{2\pi}\; \int_0^{2\pi} exp\; [\kappa\; r\; cos(\theta - \mu)]\; d\mu$$

$$L(D_i\; ; \kappa, \mu) = \frac{1}{(2\pi)^{n-1}}\; \frac{1}{\prod_{i=1}^n I_0(\kappa)}\; I_0(r\kappa)$$

The log-likelihood is then:

$$ln(L(D_i\ ;\kappa,\mu)) = log(\frac{1}{(2\pi)^{n-1}}\ \frac{1}{\prod_{i=1}^{n} I_0(\kappa)}\ I_0(r\kappa))$$

$$ln(L(\theta;\kappa,\mu)) = log(\frac{1}{(2\pi)^{n-1}}) +\ log(\frac{1}{\prod_{i=1}^{n} I_0(\kappa)}) +\ log(I_0(r\kappa))$$

$$ln(L(\theta;\kappa,\mu)) = log(1) - (n-1)\ log((2\pi)) +\ log(1) - log(\sum_{i=1}^{n} I_0(\kappa)) +\ log(I_0(r\kappa))$$

$$ln(L(\theta;\kappa)) = -(n-1)\ log((2\pi)) - log(\sum_{i=1}^{n} I_0(\kappa)) +\ log(I_0(r\kappa))$$

We construct the function `loglikelihood` in R to obtain the values from the last equation using the given data "Bees":

```
loglikelihood <- function(x,kappa) {
  n <- length(x)
  c <- -(n-1)*log(2*pi)
  r <- sqrt((sum(cos(x)))^2 + (sum(sin(x)))^2)
  b_k <- besselI(kappa, 0)
  b_rk <- besselI(r*kappa, 0)
  loglike <- c - log(sum(b_k)) + log(b_rk)
  loglike
  }
```

Based on the document in quercus we con compute the pre-normalized $U(\theta)$ as:

$$\mathfrak{U}(\theta) = exp[\ ln(\pi(\theta)\ +\ ln(L(\theta))\ -\ \max_{\theta}\{(ln(\pi(\theta)\ +\ ln(L(\theta))\}\ ]$$

We can define the prior here as:

$$\pi(\kappa,\mu) = \frac{\lambda}{2\pi} exp(-\lambda\ \kappa)$$

So, our pre-normalized $U(\theta)$ has the following form:

$$\mathfrak{U}(\theta) = exp\ [\ ln(\frac{\lambda}{2\pi} exp(-\lambda\ \kappa))\ -(n-1)\ log((2\pi))\ -\ log(\sum_{i=1}^{n} I_0(\kappa)) +\ log(I_0(r\kappa))$$
$$-\ \max_{\theta}\left\{ln(\frac{\lambda}{2\pi} exp(-\lambda\ \kappa))\ -(n-1)\ log((2\pi)) - log(\sum_{i=1}^{n} I_0(\kappa)) +\ log(I_0(r\kappa))\right\}\ ]$$

Where the log prior has the following form:

$$ln(\frac{\lambda}{2\pi} exp(-\lambda\ \kappa))$$
$$ln(\frac{\lambda}{2\pi})\ +\ ln(exp(-\lambda\ \kappa))$$
$$ln(\lambda)\ -\ ln(2\pi)\ -\ \lambda\ \kappa$$

Then, we construct our function prenorm that calculates the pre-normalized using the log-prior, log-likelihood, and given parameters and data:

```
prenorm <- function(x, kappa, lambda) {
  a <- loglikelihood(x,kappa)
  ln_prior <- log(lambda) - log(2*pi) - lambda*kappa
  a <- a + ln_prior  # add log-prior
  a <- a - max(a) # subtract maximum
  pre <- exp(a) # pre-normalized
  pre
  }
```

However, this is not enought we still need to compute the value of kappa hat, note that a simple estimate of $\kappa$ is:

$$\hat{\kappa} = \frac{\frac{r}{n}(2 - \frac{r^2}{n^2})}{1 - \frac{r^2}{n^2}}$$

Here r is defined as:

$$r = \left\{ (\sum_{i=1}^{n} cos(d_i))^2 + (\sum_{i=1}^{n} sin(d_i))^2 \right\}^{\frac{1}{2}}$$

Given all this information and the number of observations in bees (i.e. n = 279 ) we estimate $\kappa$:

$$\hat{\kappa} = \frac{\frac{\{(\sum_{i=1}^{n} cos(d_i))^2 + (\sum_{i=1}^{n} sin(d_i))^2\}^{\frac{1}{2}}}{279}(2 - \frac{\{(\sum_{i=1}^{n} cos(d_i))^2 + (\sum_{i=1}^{n} sin(d_i))^2\}}{279^2})}{1 - \frac{\{(\sum_{i=1}^{n} cos(d_i))^2 + (\sum_{i=1}^{n} sin(d_i))^2\}}{279^2}}$$

Contruct the function `kappa_hat` in order to compute the value that maximizes kappa:

```
kappa_hat <- function(x){
  n <- length(x)
  a <- sqrt((sum(cos(x)))^2 + (sum(sin(x)))^2)/n
  b <- sqrt((sum(cos(x)))^2 + (sum(sin(x)))^2)/(n^2)
  kappa_hat <- ( a *(2-b) ) / (1-b)
  kappa_hat
}
```

Using the given dataset:

```
kappa_hat(bees)
```

```
## [1] 0.0536232
```

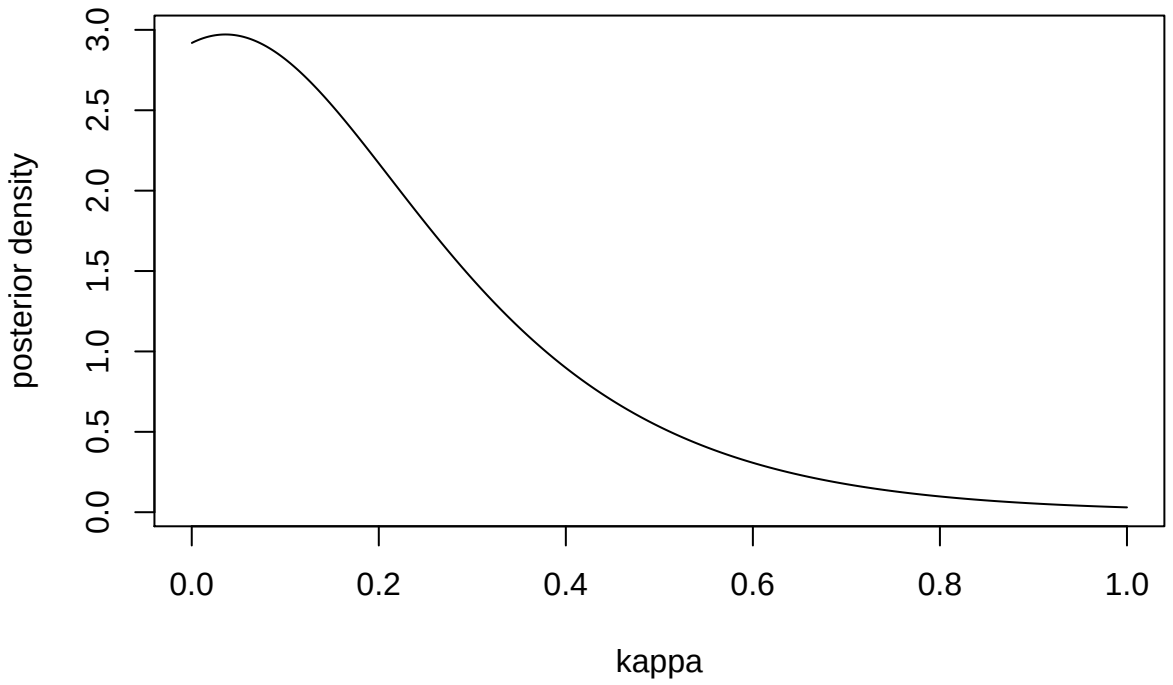
So the posterior density should be the largest for values of $\kappa$ close to 0.0536232. With this information we can set a range of values for $\kappa$ in order to construct the posterior density.

In the following R chunk we summary the functions that we constructed:

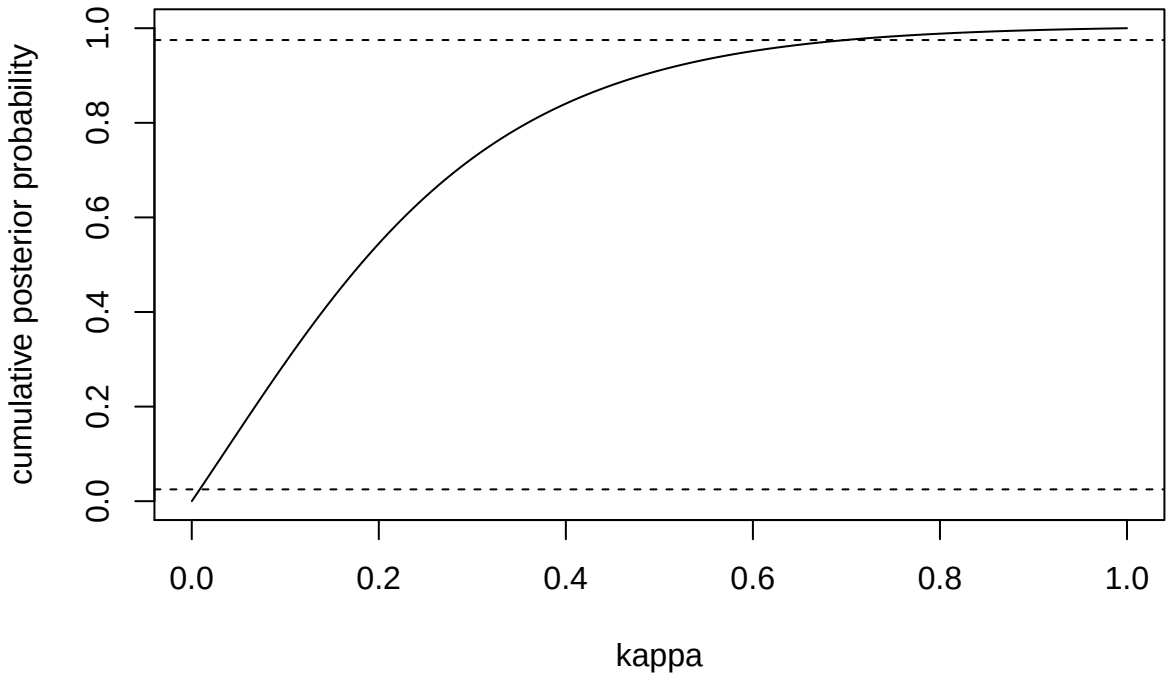
```
loglikelihood <- function(x,kappa) {
  n <- length(x)
  c <- (n-1) * log(2*pi)
  r <- sqrt((sum(cos(x)))^2 + (sum(sin(x)))^2)
```

```
  b_k <- besselI(kappa, 0)
  b_rk <- besselI(r*kappa, 0)
  loglike <- - c - log(sum(b_k)) + log(b_rk)
  loglike
}

prenorm <- function(x, kappa, lambda) {
  a <- loglikelihood(x,kappa)
  ln_prior <- (log(lambda) - log(2*pi) - (lambda*kappa))
  a <- a + ln_prior  # add log-prior
  a <- a - max(a) # subtract maximum
  pre <- exp(-a) # pre-normalized, minus to avoid problems with the numerical integration
  pre
  }

kappa_hat <- function(x){
  n <- length(x)
  a <- sqrt( (sum(cos(x)))^2 + (sum(sin(x)))^2 )/n
  b <- sqrt((sum(cos(x)))^2 + (sum(sin(x)))^2)/(n^2)
  kappa_hat <- ( a *(2-b) ) / (1-b)
  kappa_hat
}
```

```
kappa <- c(1:10000)/10000  # 10000 values of kappa
prenorm_post <- prenorm(bees, kappa = kappa , lambda = 1) # compute pre-normalized posterior
mult <- c(1/2,rep(1,9998),1/2) # multipliers for trapezoidal rule
norm <- sum(mult*prenorm_post)/10000 # integral evaluated using trapezoidal rule
post <- prenorm_post/norm # normalized posterior
plot(kappa,post,type="l",ylab="posterior density")
```



```r
post.cdf <- cumsum(mult*post)/10000 # compute the posterior cdf
plot(kappa,post.cdf,type="l",ylab="cumulative posterior probability")
abline(h=c(0.025,0.975),lty=2)
```

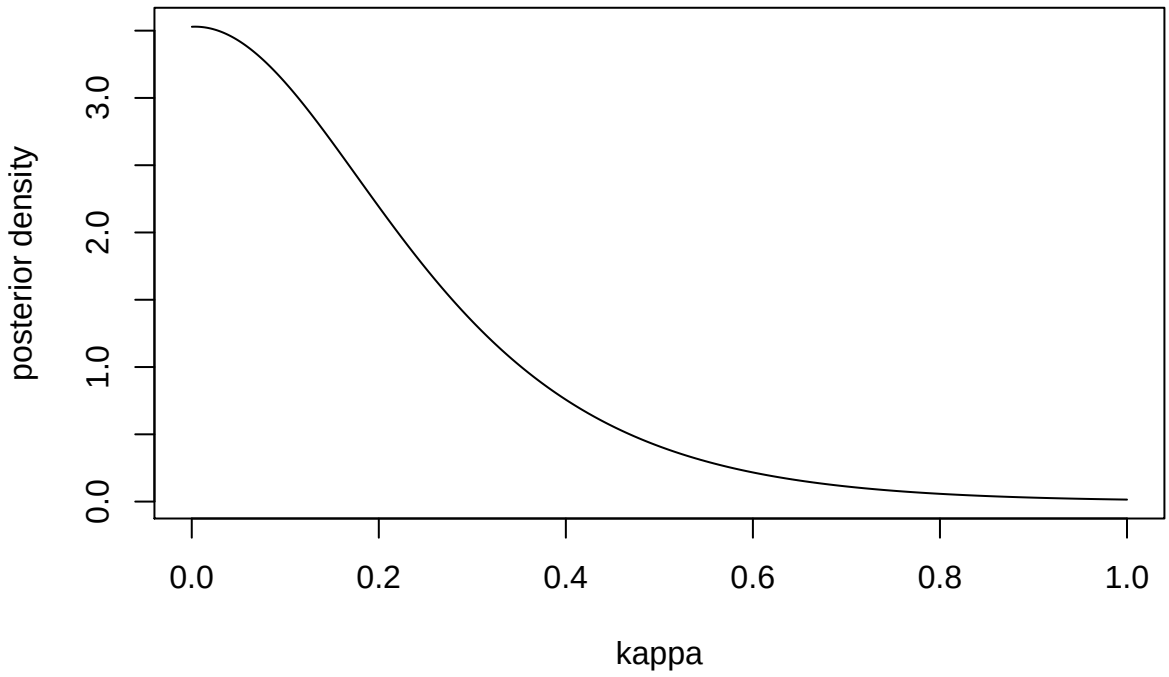



```r
lower <- max(kappa[post.cdf<0.025]) # lower limit for credible interval
upper <- min(kappa[post.cdf>0.975]) # upper limit for credible interval
c(lower,upper)
```

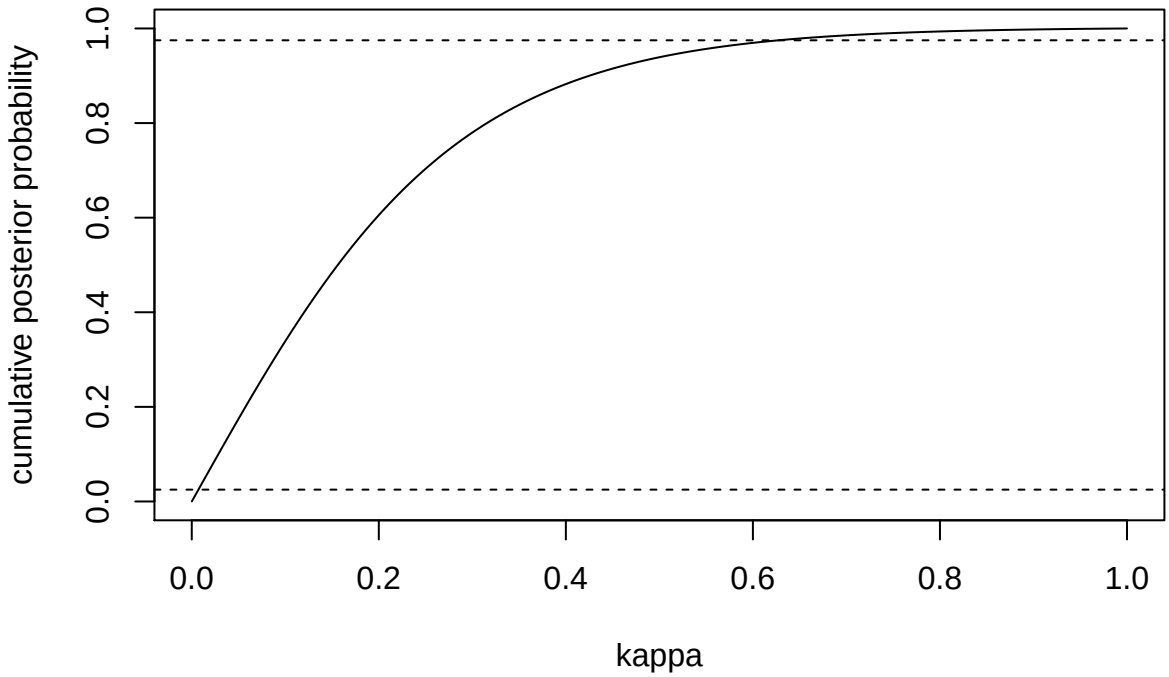
```
## [1] 0.0085 0.6989
```

Now we do the same but for Lambda = 0.1:

```r
kappa <- c(1:10000)/10000  # 671 values of kappa
prenorm_post <- prenorm(bees, kappa = kappa , lambda = 0.1) # compute pre-normalized posterior
mult <- c(1/2,rep(1,9998),1/2) # multipliers for trapezoidal rule
norm <- sum(mult*prenorm_post)/10000 # integral evaluated using trapezoidal rule
post <- prenorm_post/norm # normalized posterior
plot(kappa,post,type="l",ylab="posterior density")
```




```
post.cdf <- cumsum(mult*post)/10000 # compute the posterior cdf
plot(kappa,post.cdf,type="l",ylab="cumulative posterior probability")
abline(h=c(0.025,0.975),lty=2)
```




```
lower <- max(kappa[post.cdf<0.025]) # lower limit for credible interval
upper <- min(kappa[post.cdf>0.975]) # upper limit for credible interval
c(lower,upper)
```

```
## [1] 0.0071 0.6274
```

## Discussion of the Results

We can see that the posterior density for $\lambda = 1$ approaches zero slower than the posterior density for $\lambda = 0.1$. As mentioned before the estimated value of *kappa* that maximizes the posterior (marginal) density for $\kappa$ is close to 0.05 and we can see that the peak is constant at this point in the two posterior densities here calculated.

Moreover, the values of *kappa* and the construction of the posterior distribution here presented rule out the possibility of $\kappa = 0$ and the values of $\kappa$ are strictly positive.

As a final remark, note that the range of values of $\kappa$ here considered are [0.0001, 1], which includes the expected peak at 0.05. The credible interval for the posterior cdf when $\lambda = 1$ is [0.025, 0.975] and for the case when $\lambda = 0.1$ is [0.0071, 0.6274] which again shows that our previous claim that the density for the case when $\lambda = 0.1$ approaches zero more rapidly than the case when $\lambda = 1$.

(d) The model that we've used to this point assumes that $\kappa$ is strictly positive and thus rules out the case where $\kappa = 0$ (which implies that $D_1, \cdots, D_n$ are uniformly distributed). We can actually address the general model (i.e. $\kappa \geq 0$) by putting a prior probability $\theta$ on the model $\kappa = 0$ (which implies that we put a prior probability $1 - \theta$ on $\kappa > 0$). The prior distribution on the parameter space $\{(\kappa, \mu) : \kappa \geq 0, 0 \leq \mu < 2\pi\}$ has a point mass of $\theta$ at $\kappa = 0$ and the prior density over $\{(\kappa, \mu) : \kappa > 0, 0 \leq \mu < 2\pi\}$ is $(1 - \theta)\pi(\kappa, \mu)$. (This is a simple example of a **"spike-and-slab"** prior, which are used in Bayesian model selection; in this case, the "spike" refers to the prior point mass $\theta$ and the "slab" is the prior density on $(\kappa, \mu)$ for $\kappa > 0$.)
The posterior probability that $\kappa = 0$ is then

$$\text{Prob}(\kappa = 0 | d_1, \cdots, d_n) = \frac{\theta(2\pi)^{-n}}{\theta(2\pi)^{-n} + (1 - \theta) \int_0^\infty \int_0^{2\pi} \mathcal{L}(\kappa, \mu)\pi(\kappa, \mu)\, d\mu\, d\kappa}$$

Using the prior $\pi(\kappa, \mu)$ in part (b) with $\lambda = 1$, evaluate $\text{Prob}(\kappa = 0 | d_1, \cdots, d_n)$ for $\theta = 0.1, 0.2, 0.3, \cdots, 0.9$. (This is easier than it looks as you've done much of the work in part (c).)

Address the general model (i.e $\kappa \geq 0$) by putting a prior probability on the model $\kappa = 0$ (this implies a prior probability on $1 - \theta$ on $\kappa > 0$). The prior density over $\{(\kappa, \mu)$: $\kappa > 0,\ 0 \leq \mu < 2\pi\}$ is $(1 - \theta)\pi(\kappa, \mu)$, where

$$\pi(\kappa, \mu) = \frac{\lambda}{2\pi}\ exp(-\lambda\ \kappa)$$

this last prior is used with $\lambda = 1$ and becomes:

$$\pi(\kappa, \mu) = \frac{1}{2\pi}\ exp(-1\ \kappa)$$

So, the prior for this part is:

$$(1 - \theta)\ \pi(\kappa, \mu) \equiv (1 - \theta)\ \frac{1}{2\pi}\ exp(-1\ \kappa)$$

Note that the new log prior looks like:

$$\begin{aligned} & ln((1 - \theta)\frac{1}{2\pi} exp(-1\ \kappa)) \\ ln(1 - \theta) + ln(\frac{1}{2\pi}) & + ln(exp(-1\ \kappa)) \\ ln(1 - \theta) & - ln(2\pi) - 1\ \kappa \end{aligned}$$

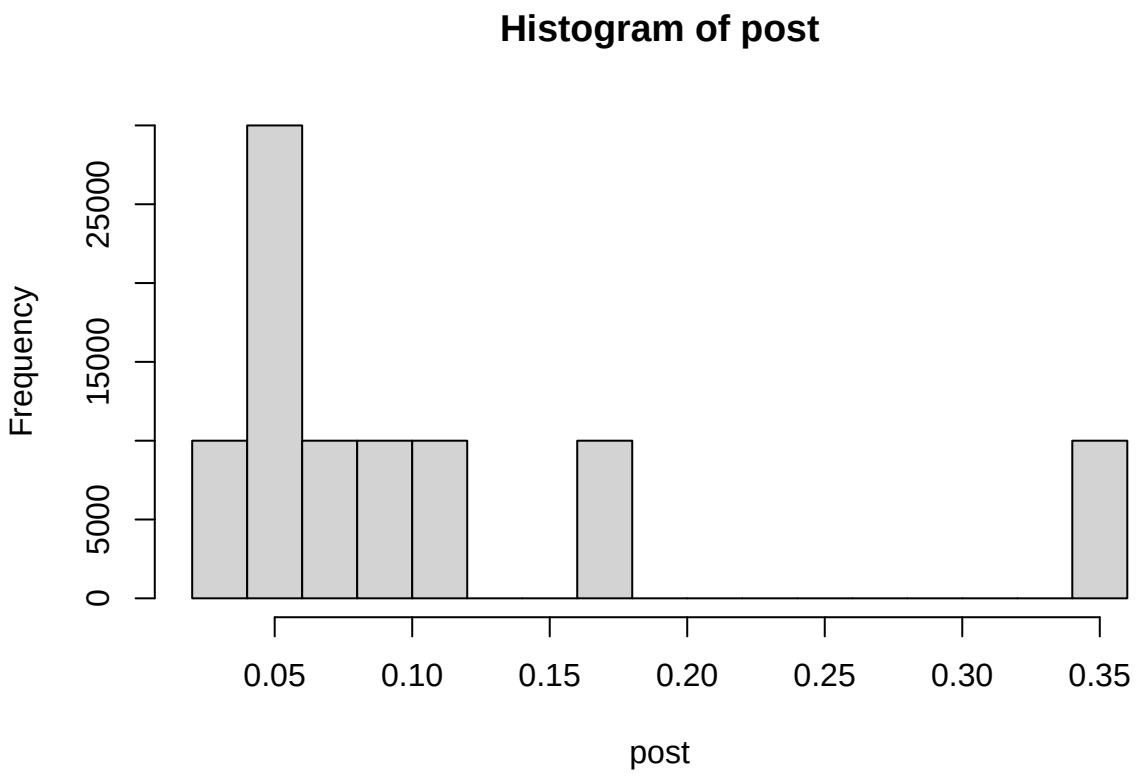
Using the code from part c:

```
loglikelihood2 <- function(x,kappa) {
  n <- length(x)
  c <- (n-1) * log(2*pi)
  r <- sqrt((sum(cos(x)))^2 + (sum(sin(x)))^2)
  b_k <- besselI(kappa, 0)
  b_rk <- besselI(r*kappa, 0)
```

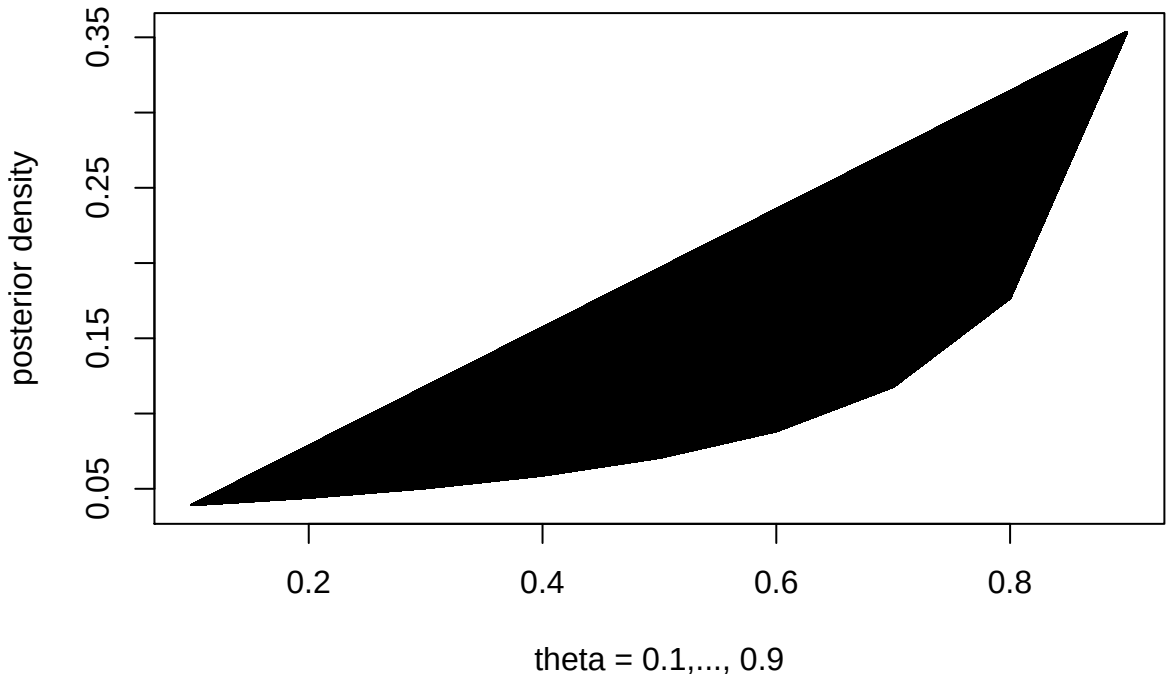
```
  loglike <- - c - log(sum(b_k)) + log(b_rk)
  loglike
}

prenorm2 <- function(x, kappa, lambda, theta) {
  a <- loglikelihood2(x,kappa)
  ln_prior <- log(1-theta) + (log(lambda) - log(2*pi) - (lambda*kappa))
  a <- a + ln_prior  # add log-prior
  a <- a - max(a) # subtract maximum
  pre <- exp(-a) # pre-normalized, minus to avoid problems with the numerical integration
  pre
}
```

*For all thetas*

```
#Kappa = 0
theta <- seq(from = 0.1, to = 0.9, by = 0.1)
prenorm_post <- prenorm2(bees, kappa = 0 , lambda = 1, theta = rep(theta,10000)) # compute pre-normaliz
mult <- c(1/2,rep(1,9998),1/2) # multipliers for trapezoidal rule
norm <- sum(mult*prenorm_post)/10000 # integral evaluated using trapezoidal rule
post <- prenorm_post/norm # normalized posterior
hist(post)
```




```
plot(rep(theta,10000),post,type="l",ylab="posterior density", xlab="theta = 0.1,..., 0.9")
```


posterior density
0.05
0.15
0.25
0.35
0.2
0.4
0.6
0.8
theta = 0.1,..., 0.9

(c) Consider the following model:

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (i = 1, \cdots, n)$$

where $x_i = i/n$ and $\{\varepsilon_i\}$ are independent $\mathcal{N}(0,1)$ random variables. For least squares estimation, the variance of $\widehat{\beta}_1$ tends to 0 like constant$/n$ as $n \to \infty$. For LMS estimation, $\text{Var}(\widehat{\beta}_1) \approx \gamma/n^\alpha$ for some $\gamma > 0$ and $\alpha > 0$. In lecture, we claimed that $\alpha = 2/3$. The theoretical proof of this is very technical; however, it is possible to estimate $\alpha$ via simulation.

The idea here is very simple: We can compute $\widehat{\beta}_1$ based on $n$ observations and replicate this process $M$ times, which results in $M$ values of $\widehat{\beta}_1$. We can then estimate $\text{Var}(\widehat{\beta}_1)$ from these $M$ values, in which case

$$\begin{aligned} \widehat{\text{Var}}(\widehat{\beta}_1) &\approx \gamma/n^\alpha \\ \text{or} \quad \ln\left(\widehat{\text{Var}}(\widehat{\beta}_1)\right) &\approx \ln(\gamma) - \alpha \ln(n) \end{aligned}$$

Repeating this process for a range of sample sizes allows us estimate $\alpha$.
Estimate $\alpha$ based on sample sizes $n = 50, 100, 500, 1000, 5000$.

Consider the following model

$$Y_I = \beta_0 + \beta_1 x_i + \epsilon_i, \qquad (i = 1, ..., n)$$

where: $x_i = \dfrac{i}{n}$ *and* $\epsilon_i$ are independent $\mathcal{N}(0,1)$ random variables.

For LMS estimation

$$Var(\hat{\beta}_1) \approx \gamma/n^\alpha$$

for some $\gamma > 0$ and $\alpha > 0$. In class we claimed that $\alpha = 1/3$, here estimate $\alpha$ via simulation.

Step 1) compute $\hat{\beta}_1$ based on n=50 obs

Step 2) Replicate this process M times (obtaining M values of $\hat{\beta}_1$).

Step 3) Estimate

$$Var(\hat{\beta}_1)$$

from these M values in which case:

$$\begin{aligned} \hat{Var}(\beta_1) &\approx \frac{\gamma}{n^\alpha} \\ \text{or:} \quad ln(\hat{Var}(\beta_1)) &\approx ln(\gamma) - \alpha\ ln(n) \end{aligned}$$

*Using a sample size of 50 (i.e: $n = 50$ ):*

```
library(MASS)
n <- 50
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  y <- rnorm(n, 0, 1)
  r <- lmsreg(y~x)
```

```
  beta <- c(beta,r$coef[2])
}
beta_50 <- beta
save(beta_50, file = "beta_50.RData")
```

```
load("beta_50.RData")
var50 <- var(beta_50)
var50
```

```
## [1] 1.388859
```

*Using a sample size of 50 (i.e: $n = 100$ ):*

```
library(MASS)
n <- 100
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  y <- rnorm(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_100 <- beta
save(beta_100, file = "beta_100.RData")
```

```
load("beta_100.RData")
var100 <- var(beta_100)
var100
```

```
## [1] 0.8161859
```

*Using a sample size of 50 (i.e: $n = 500$ ):*

```
library(MASS)
n <- 500
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  y <- rnorm(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_500 <- beta
save(beta_500, file = "beta_500.RData")
```

```
load("beta_500.RData")
var500 <- var(beta_500)
var500
```

```
## [1] 0.210944
```

*Using a sample size of 50 (i.e: $n = 1000$ ):*

```
library(MASS)
n <- 1000
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  options(mc.cores=parallel::detectCores())
  y <- rnorm(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_1000 <- beta
save(beta_1000, file = "beta_1000.RData")
```

```
load("beta_1000.RData")
var1000 <- var(beta_1000)
var1000
```

```
## [1] 0.1419492
```

*Using a sample size of 50 (i.e: $n = 5000$ ):*

```
library(MASS)
n <- 5000
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  options(mc.cores=parallel::detectCores())
  y <- rnorm(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_5000 <- beta
save(beta_5000, file = "beta_5000.RData")
```

```
load("beta_5000.RData")
var5000 <- var(beta_5000)
var5000
```

```
## [1] 0.04482628
```

Summarizing the variances obtained:

```
cbind(var50, var100, var500, var1000, var5000)
```

```
##          var50    var100   var500    var1000    var5000
## [1,] 1.388859 0.8161859 0.210944 0.1419492 0.04482628
```

Now we only need to estimate $\alpha$, we can do this using a simple OLS estimation:

$$ln(\hat{Var}(\beta_1)) \approx ln(\gamma) - \alpha\ ln(n)$$

So we take the logarithm of our estimated variances:

```
y <- log(c(var50, var100, var500, var1000, var5000))
log(rbind(c(var50, var100, var500, var1000, var5000)))
```

```
##           [,1]       [,2]      [,3]      [,4]      [,5]
## [1,] 0.3284828 -0.2031131 -1.556163 -1.952286 -3.104961
```

This is corresponding for each log-value of n, that is:

```
##           n50    n100     n500    n1000    n5000
## [1,] 3.912023 4.60517 6.214608 6.907755 8.517193
```

So, we run our regression as follows:

```
library(MASS)
ols_reg <- lm(y~x)
ols_reg <- ols_reg$coefficients
lms <- lmsreg(y~x)
lms_reg <- lms$coefficients
```

Note that we run the OLS regression and LMS regressionjust to comparison purposes. The estimates for the coefficients that these regressions produce are very similar:

```
##               ols_reg    lms_reg
## (Intercept)  3.221948  3.3049219
## x           -0.749344 -0.7613393
```

Finally, our estimate of $\alpha$ is **0.749** for the OLS regression and **0.7612** for the LMS regression, this is based on samples sizes of n = 50, 100, 500, 1000, 5000 and the assumption that the errors are independent N(0,1) random variables.

(d) Repeat part (c) using Cauchy errors. (For Cauchy errors, the variance of the least squares estimator does not tend to 0 as $n \to \infty$.) Do you get a similar value of $\alpha$?

```
library(MASS)
n <- 50
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  y <- rcauchy(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_50_cauchy <- beta
save(beta_50_cauchy, file = "beta_50_cauchy.RData")
```

```
load("beta_50_cauchy.RData")
var50_cauchy <- var(beta_50_cauchy)
var50_cauchy
```

```
## [1] 1.292033
```

```
library(MASS)
n <- 100
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  y <- rcauchy(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_100_cauchy <- beta
save(beta_100_cauchy, file = "beta_100_cauchy.RData")
```

```
load("beta_100_cauchy.RData")
var100_cauchy <- var(beta_100_cauchy)
var100_cauchy
```

```
## [1] 0.663055
```

```
library(MASS)
n <- 500
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  y <- rcauchy(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
```

```
}
beta_500_cauchy <- beta
save(beta_500_cauchy, file = "beta_500_cauchy.RData")
```

```
load("beta_500_cauchy.RData")
var500_cauchy <- var(beta_500_cauchy)
var500_cauchy
```

```
## [1] 0.1980633
```

```
library(MASS)
n <- 1000
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  options(mc.cores=parallel::detectCores())
  y <- rcauchy(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_1000_cauchy <- beta
save(beta_1000_cauchy, file = "beta_1000_cauchy.RData")
```

```
load("beta_1000_cauchy.RData")
var1000_cauchy <- var(beta_1000_cauchy)
var1000_cauchy
```

```
## [1] 0.1247424
```

```
library(MASS)
n <- 5000
nrep <- 1000
x <- c(1:n)/n
beta <- NULL
for (i in 1:nrep) {
  options(mc.cores=parallel::detectCores())
  y <- rcauchy(n, 0, 1)
  r <- lmsreg(y~x)
  beta <- c(beta,r$coef[2])
}
beta_5000_cauchy <- beta
save(beta_5000_cauchy, file = "beta_5000_cauchy.RData")
```

```
load("beta_5000_cauchy.RData")
var5000_cauchy <- var(beta_5000_cauchy)
var5000_cauchy
```

```
## [1] 0.04403548
```

We summarized the estimated variances for $\beta_1$ when using the cauchy errors:

```
cbind(var50_cauchy, var100_cauchy, var500_cauchy, var1000_cauchy, var5000_cauchy)
```

```
##      var50_cauchy var100_cauchy var500_cauchy var1000_cauchy var5000_cauchy
## [1,]     1.292033      0.663055     0.1980633      0.1247424     0.04403548
```

In log-terms we have:

```
y_c <- log(c(var50_cauchy, var100_cauchy, var500_cauchy, var1000_cauchy, var5000_cauchy))
log(cbind(var50_cauchy, var100_cauchy, var500_cauchy, var1000_cauchy, var5000_cauchy))
```

```
##      var50_cauchy var100_cauchy var500_cauchy var1000_cauchy var5000_cauchy
## [1,]    0.2562169    -0.4108974     -1.619169      -2.081505       -3.12276
```

And using the same approach as part c we have:

```
library(MASS)
ols_reg <- lm(y_c~x)
ols_reg_c <- ols_reg$coefficients
lms <- lmsreg(y_c~x)
lms_reg_c <- lms$coefficients
```

So, the estimated coefficients for $\alpha$ and $\gamma$ are summarized in the following table:

```
##              ols_reg_c  lms_reg_c
## (Intercept)  2.9986972  2.4341235
## x           -0.7285798 -0.6530013
```

When the errors are assumed to follow a Cauchy distribution the estimates for $\alpha$ differ from the case when the errors are assumed to come from a independently N(0,1) distribution. Then, in the case of the OLS regression the estimate for $\alpha$ is **0.7286** and when using the LMS regression the estimate for $\alpha$ is **0.653** which is clearly a slightly different estimate than when using the OLS regression. This is based on samples sizes of n = 50, 100, 500, 1000, 5000 and the assumption that the errors are independent N(0,1) random variables.

So, we conclude that the values for $\alpha$ when using Cauchy errors are slightly different than part c, we remark that for Cauchy errors the variance of the OLS estimator does not tend to zero as n gets large, this explains the difference in the estimates.